UDC 004.9 **doi: 10.26906/SUNZ.2025.4.121-125**

Viacheslav Radchenko, Yuliia Andrusenko

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

INTELLIGENT APPROACH TO PLANNING TAKING INTO ACCOUNT THE CONCEPT OF ACCEPTABLE WORK BALANCE

Abstract. The article presents an intelligent approach to the problem of scheduling computing tasks in a distributed environment taking into account the concept of admissible load balance. The proposed model combines the principles of multi-criteria optimization and machine learning for adaptive resource allocation between system nodes. The introduction of an admissibility coefficient allows you to dynamically limit the set of machines available for performing a specific task, which increases the efficiency of capacity use and reduces task waiting time. The developed approach takes into account various efficiency criteria — productivity, load balance, and scheduling stability — and can be implemented in grid and cloud infrastructures. Experimental results demonstrate that the use of intelligent algorithms allows you to achieve an optimal compromise between system performance and uniform loading of computing resources.

Keywords: planning, distributed systems, grid, resources, heterogeneous systems.

Introduction

Modern distributed and hybrid computing systems are characterized by high dynamic loads and heterogeneous resources. In such conditions, the problem of effective work scheduling becomes particularly relevant, since incorrect task assignment leads to irrational use of capacity and a decrease in overall system performance. Traditional scheduling methods, which are based on static algorithms or heuristics, often do not take into account the variability of the environment and user priorities, which limits their effectiveness. One of the reasons for the inefficiency of online scheduling is the situation when small tasks occupy large computing nodes, blocking the execution of parallel or resource-intensive processes. To avoid such scenarios, it is advisable to use the concept of admissible work balance, which introduces an admissibility parameter when choosing a set of machines. This allows you to adaptively adjust resources in accordance with current requirements and reduce the risk of system overload.

Modern approaches to planning are increasingly based on artificial intelligence methods, in particular machine learning and evolutionary algorithms, which provide the system with the ability to self-learn and make decisions under uncertainty. This approach allows for greater consistency between the criteria of time, productivity and energy efficiency, which is especially important for cloud and grid infrastructures.

1 Research objective

The objective of this research is to develop an intelligent model of work planning in a distributed environment based on the concept of admissible balance, which ensures effective load distribution between computing resources. To achieve this goal, it is planned to:

– analyze existing planning methods in grid and cloud systems;

– determine a mathematical model of an admissible set of machines using the admissibility parameter;

– develop an adaptive decision-making algorithm using machine learning methods;

– conduct an experimental comparison of the effectiveness of the proposed approach with traditional planning methods.

The implementation of this goal will contribute to increasing the efficiency of resource use in high-performance computing environments and forming the basis for creating intelligent automated load management systems.

Scheduling is a crucial aspect of achieving high performance in distributed systems. Various scheduling algorithms have already been proposed and implemented in various types of systems. However, many unanswered questions remain in this area. One of the main challenges is developing coordinated resource allocation mechanisms that enable more efficient utilization of resources and improve service quality. In general, the problem of job scheduling on multiprocessors is well understood and has been the subject of decades of research, addressing theoretical aspects or hints for implementing real-world systems. Scheduling in GRID, on the other hand, is almost exclusively considered by practitioners seeking suitable implementations. The highest level consists of a GRID scheduler (a resource broker, or meta-scheduler), which receives job requests and distributes jobs to the appropriate GRID resource. Managing a specific resource is the responsibility of a local management system, which knows the current state of its machine and the jobs assigned to it. Local scheduling is applied independently to each machine. At each level, various constraints and specifications are taken into account. Here, we consider parallel jobs that have a specified degree of parallelism and during which only a specified number of processors or cores must be assigned to them.

2 Literature Review

A feasible assignment policy, which excludes certain machines from the set of machines available for assigning a given job, was proposed in [1]. In [2], it was shown that the competitive factor of the MLBa+PS algorithm ranges from 17 to infinity with varying the feasibility factor. In [3], the existing results of [2] are improved and expanded, achieving an approximation factor of 9 for the offline case and a competitive factor of 11 for the online case. An approximation factor of 3 and a competitive factor of 5 are also obtained for cases where all slaves converge on the smallest machine, as in

the problem discussed in [4]. To demonstrate the practicality and competitiveness of the algorithms, a comprehensive performance study using simulations is conducted in [5]. Workloads based on real production systems are used to examine three GRID scenarios based on heterogeneous HPC systems and to study the problem of job scheduling with uncertain execution time requirements, as in a real runtime environment, only user-provided job execution time estimates are available. The algorithms are focused on parallel, computationally intensive jobs and make scheduling decisions without precise performance information, particularly regarding job execution time. They are simple, operate on a job-by-job basis, and allow for efficient implementation in real systems.

3 Acceptable distribution of work

The online scheduling problem is defined as follows: n parallel jobs J_1, J_2, …, J_n must be assigned to parallel machines N_1, N_2, …, N_m. Let m_i will be the number of identical processors in the machine N_i, also known as the size of the machine N_i. Let $s_{f,l} = \sum_{i=f}^{l} m_i$ will be the total number of processors belonging to machines from N_f , to N_1. Let us assume, without loss of generality, that parallel machines are arranged in non-decreasing order of size. $m_1 \le \ldots \le m_m$.

Every job j_j is described by a set (r_j, $size_j$, p_j , p'_j), determining the following characteristics of the operation: the time of its occurrence $r_j \ge 0$ – time relative to the start of the schedule, its size $1 \le size_j \le m_m$ – processor requirements, execution time p_j and assessment of user execution time p_j.

In an online scenario, no job parameters are available until the job is submitted. Planning is difficult if the processing time of a job is only known after it has been completed. This is called a non-clear-run scenario. In some real-world systems, the user must provide an estimate of the processing time for their job to prevent wasting computing resources when working with programs that contain errors, such as infinite loops. This estimate can also be used by the scheduler, although estimates can be quite inaccurate.

Let $w_j = p_j\ sizej$ this is the amount of work j_j , also referred to as its schedule area or its consumed resource. Jobs are submitted over time and must be immediately and definitively assigned to a single machine. However, the allocation of processors for a job may be delayed until the required number of processors is actually available. The job is executed in space-partitioning mode by allocating $size_j$ processors for an uninterrupted period of time p_j . TSince neither interruptions nor multi-machine execution nor shared distribution of processors from different machines are permitted, the work j can be performed on the machine N_i only if $size_j\ i$.

Time of completion j_j copy in schedule S is denoted by $c_j = (S, I)$. Formally, the duration of work in schedule S and copy I is equal to $C_{max}(S, I)$. The optimal time for the end of work of instance I is indicated by $C_{max}(S, I)$. Further on, wherever possible without causing ambiguity, we will omit the instance and schedule S. Let us denote the GRID machine model as GP_m. In three-field notation α | β | γ, our planning task is characterised as $GP_m|r_j, size_j|C_{max}$ for the optimisation criterion C_{max}. Target functions are based on metrics, and notation is used to denote this problem MPS (Multiple machine Parallel Scheduling.

A local resource management system can use various scheduling algorithms. In the experimental sections of this chapter, it is assumed that the LRMS uses an online parallel scheduling algorithm: First-Come-First-Serve policy with the EASY backfilling algorithm, where the scheduler can use later tasks to fill gaps in the schedule, even if this delays the expected start time of other tasks, as long as the expected start time of the first task is not delayed. To apply EASY backfilling, the user-specified estimated execution time is used. Formally, the competitive factor of algorithm A is defined as $\rho_A = max_I \frac{C_{max}(S_A,I)}{C^*_{max}(I)}$ I for all possible instances of the problem. Again, we omit algorithm A if this does not cause ambiguity. The approximation factor is determined similarly for deterministic offline planning problems.

It should be noted that in our deterministic planning, all tasks are available at the beginning and the planner knows the parameters of all tasks. It is assumed that the resources involved are stable and designed to work in GRID. Well-known metrics for algorithm performance are also considered, typically used to denote the goals of various participants in GRID planning: end users, local resource providers, and administrators:

$$SD_b = \frac{1}{n} \cdot \sum_{j=1}^{n} \frac{c_j - r_j}{\max\left(10; p_j\right)} \tag{1}$$

and average waiting time

$$t_w = \frac{1}{n} \cdot \sum_{j=1}^{n} \left(c_j - r_j - p_j\right). \tag{2}$$

To eliminate the impact of very short jobs, i.e., those with close to zero execution time, the slowdown is limited to a commonly used threshold of 10 s.

This work uses the waiting time tw instead of the response time:

, $$TA = \frac{1}{n} \cdot \sum_{j=1}^{n} \left(c_j - r_j\right). \tag{3}$$

and the sum of the waiting times of the jobs

$$SWT = \sum_{j=1}^{n} \left(c_j - r_j - p_j\right). \tag{4}$$

The differences between these metrics are constant regardless of the scheduler used:

$$const = \frac{1}{n} \sum_{j=1}^{n} p_j. \tag{5}$$

Resource-based metrics such as utilization are not used

$$U = \sum_{j=1}^{n} \frac{p_j \cdot size_j}{c_{\max} \cdot \max\left(s_j; m\right)} \tag{6}$$

and throughput $Th = n / c_{\max}$. (7)

4 Hierarchical planning system

Job scheduling is a critical part of the efficient operation of computing systems. Various aspects of the problem are discussed in the literature to address emerging challenges facing distributed systems: centralized, hierarchical, and distributed models; static and dynamic scheduling policies [6]; multi-objective optimization [7]; adaptive policies related to the dynamic behavior of resources [8]; autonomous control; QoS constraints; economic models; resource selection; scheduling of data and resource-intensive computations; workflow scheduling; data locality, resource bindings for execution and data storage [9]; replication; performance evaluation.

Theoretical evaluation of scheduling in distributed systems is studied to provide hints for the implementation of real systems in which only user estimates of job execution times may be available.

Therefore, it is important that the scheduler be able to integrate user estimates and available information with dynamic and static resource allocation strategies. This is one of the goals of this chapter. Distributed systems vary significantly in size, and their workload is very dynamic. Quality of service is an important performance characteristic for scheduling algorithms. Therefore, worst-case theoretical analysis is a relevant approach, as it provides such guarantees in systems with non-stationary parameters.

In this article, we consider an algorithm that knows nothing about jobs other than the number of outstanding jobs in the system and their processor requirements.

Scheduling algorithms for two-level GRID models can be divided into a global part of job allocation and a local part of job scheduling. Thus, MPS is considered a two-stage scheduling strategy: MPS = MPS_Alloc + PS. In the first stage, each job is assigned to a suitable machine using a specified selection criterion. In the second stage, the PS algorithm is applied to each machine for the jobs assigned in the previous stage. It is easy to see that the MPS algorithm's contention factor is bounded below by the PS algorithm's contention factor, given a degenerate GRID that contains only one machine. The best possible online non-clairvoyant PS algorithm (with unknown job execution time) has a contention factor of 2, where denotes the number of processors on a single parallel machine.

Three levels of information are available for job allocation. Each level differs in the type and amount of information required to generate a schedule (Table 1).

Table 1 – **Scheduling Strategies**

Strategy	Level	Description
Random	1	Distributes work to the machine randomly
ML_P	1	Distributes job j to the machine with the lowest CPU load at time r_j
MPL	1	Distributes job j to the machine with the lowest CPU requirements at time r_j
LB_{al}_S	1	Allocates work to the machine with the smallest standard deviation of CPU requirements, taking into account all machines when work is assigned to it $\min_{q=1..m} \sqrt{\sum_{i=1}^{m}\left(PL_i^q - \overline{PL}\right)^2 / m}$
MLB	2	Distributes work to the machine with the lowest CPU load at the moment r_j
MCT	3	Allocates work to the machine with the earliest completion time { C_{max} }
MWWT_S	3	Distributes work to the machine with the minimum weighted average waiting time
MST	3	Assigns a job to the machine with the earliest start time for that job

5 Acceptable distribution of work

The problem of scheduling jobs on distributed machines online has rarely been addressed so far. Unfortunately, in the worst case, this can lead to inefficient use of the entire system. One of the structural causes of such inefficiencies in online scheduling is the potential use of large machines by jobs with small processor requirements, forcing highly parallel jobs to wait until completion if they are submitted later.

Acceptable set of machines for work j_j are the machines with indexes $\{f_j, \ldots, l_j\}$ or simply $\{f, \ldots, l\}$, if it does not cause ambiguity, where f_j - smallest index i such that $m_i \geq size_j$, and l_j smallest machine index i (рис.1), where s_{f_j,l_j}, the total number of processors belonging to machines from N_{f_j} to N_{l_j}. It should be noted that always j. The admissibility coefficient parameterizes the admissibility of machines used for job distribution. Note that at least one machine is admissible, since is strictly greater than 0, while specifies that all available machines are admissible. If is the smallest index, for example, $size_j$, then the work can be distributed among machines from index to . The admissibility coefficient reduces the available machines to machines with indices from f to 1 (Fig. 1).

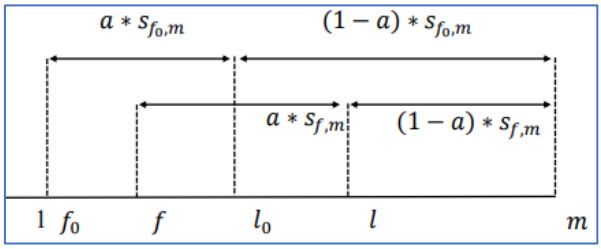

Fig. 1. Example of acceptable machines for assignment of work with factor α

Assume that the work J_j allocated to the machine from the set f,…,*l* containing processors α $s_{f,m}$ (Fig. 1). Therefore, (1–α) $s_{f,m}$ processors are excluded from $s_{f,m}$ processors available for J_j. Note that the machines are indexed in order of size, not decreasing. Obviously, the total set of machines is represented by a whole set. 1, …, m. The load planning problem is a multi-objective problem, considering

several performance criteria. Resource providers and users often have different, sometimes conflicting, goals, ranging from minimizing response time to optimizing resource utilization. Grid resource management can utilize multi-objective decision support. A general multi-objective decision methodology based on Pareto optimality can be applied to this purpose. However, achieving a fast solution using Pareto dominance, as is necessary for resource management, is very difficult. The problem is often simplified to a single-objective problem or to various methods for combining objectives. There are various ways to model preferences; for example, they can be explicitly specified by stakeholders to indicate the importance of each criterion or the relative importance between criteria. This can be done by defining criterion weights or ranking criteria by importance.

The goal is to find a robust and well-performing strategy across all test cases, with the expectation that it will also perform well under other conditions, such as different configurations and workloads. The analysis is conducted as follows. First, the degradation—the relative performance error—of each strategy for each metric is estimated. This is done relative to the best performance for that metric:

$$MTR = 100 \cdot (metric/best\_metric) - 1. \quad (7)$$

Thus, for the three main metrics, each strategy is characterized by three 9-digit numbers for 3 grids, reflecting its relative performance degradation in test cases. These three values are then averaged, assuming equal importance for each metric, and the strategies are ranked for each grid. The best strategy, with the smallest average performance degradation, has a rank of 1; the worst strategy has a rank of 9. The average performance degradation is then calculated for G d1, G d2, and G d3. The main goal is to identify strategies that perform reliably in different scenarios; that is, to try to find a compromise that takes into account all of our test cases.

6 Assignments operate in divisions of the system with parameter α

Let us have 5 machines with a different number of processors: 2,4, 8, 16, 32. Machines sorted by size ($s_1 \le s_2 \le s_3 \le s_4 \le s_5$). Jobs with different requirements for the number of processors enter the system, let's define their flow: 2, 8, 20. Let us determine the permissible set of machines, for each job we will determine the minimum index j_l so that s_j ≥ size_j (Table 2).

Table 2 – **The permissible set of machines**

Work	$size_j$	j_l	Suitable machines (f,…,l)	s_{f,m}	α = 0.5	Available processors (α·s_{f,m})
J_1	2	1	{M_1, M_2, M_3, M_4, M_5}	62	0.5	31
J_2	8	3	{M_3, M_4, M_5}	56	0.5	28
J_3	20	4	{M_4, M_5}	48	0.5	24

The parameter α = 0.5 means that only 50% of the processors in the allowable range are available for assignment.

• Job J_1 can only be assigned to a subset of machines, with a total of 31 processors.

For example, J_1 will get machine M_2 (4 processors).

• Job J_2 is assigned to M_3 (8 processors), because this is enough for its requirements.

• Job J_3 needs 20 processors, so it chooses M_4 (16) + part of M_5 (4 processors).

In the online assignment situation, let's analyze the efficiency:

• If J_1 (a small job) arrives before J_3 and occupies a large machine (e.g. M_4), then J_3 is forced to wait, even though it needs more power.

• This leads to inefficient use of the system.

Three main metrics can be considered to evaluate the effectiveness of allocation strategies:

1. Waiting time (T_1)
2. Resource utilization (U)
3. Load balance (L)

For each strategy, the relative degradation of each metric compared to the best result is calculated.

For example:

Strategy	ΔT_1	ΔU	ΔL	Average decrease	Rank
S_1	0.05	0.10	0.08	0.076	2
S_2	0.02	0.12	0.05	0.063	**1**
S_3	0.10	0.18	0.09	0.123	3

Strategy S_2 receives rank 1 because it has the smallest average performance degradation in the three test grid scenarios, the α-parameter limits the set of machines available for work and helps to avoid overuse of large nodes by small tasks. In online mode, incorrect assignment leads to resource blocking. Multi-criteria analysis (Pareto method) allows you to choose a scheduling strategy that works stably in different system configurations**.**

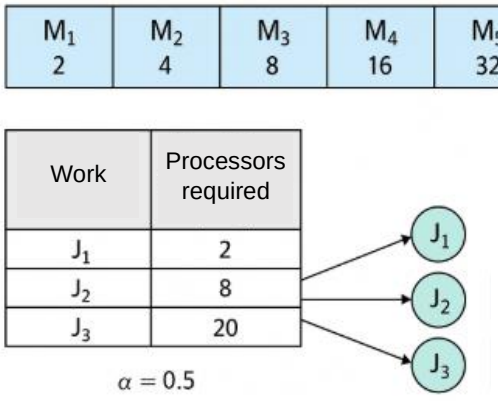

Fig. 2. Example of permissible machines for assigning jobs by factor α

The presented diagram (Fig. 2) demonstrates an example of permissible distribution of jobs between

machines in a distributed system with a permissiveness coefficient α = 0.5. It shows that only a part of the machines from the entire resource pool is available for a specific job depending on its processor requirements ($size_j$). According to the model, smaller jobs (for example, J_1) can be performed on several small machines, while more resource-intensive jobs (J_2, J_3) have a limited set of possible machines due to the parameter α. This provides a balance between the system load level and the efficiency of processor use.

Conclusions

The graphical analysis allows us to draw the following conclusions:

• using the coefficient α allows us to avoid excessive use of large machines by small jobs;

• the approach helps to reduce downtime of high-performance nodes;

• the system achieves better load balancing with different intensities of incoming jobs;

• in the case of online scheduling, this method ensures stability and uniformity of distribution under changing conditions.

Therefore, the use of the admissibility coefficient α in the work assignment model is an effective way to optimize the use of computing resources in hybrid or grid environments.

СПИСОК ЛІТЕРАТУРИ

1. Jothi, G., and Saravanan, P. (2017), "A New Algorithm to Find the Optimal Feasible Assignment for an Assignment Problem", Int. journal of engineering research & technology, vol. 5, issue 04, doi: https://doi.org/10.17577/IJERTCONV5IS04013
2. Kristensen, J.T., Valdivia, A. and Burelli, P. (2020), "Estimating player completion rate in mobile puzzle games using reinforcement learning", Proc. of the IEEE Conference Computational Intelligence and Games, pp. 636–639, doi: https://doi.org/10.1109/CoG47356.2020.9231581
3. Zhu, W. and Rosendo, A. (2021), "A functional clipping approach for policy optimization algorithms". IEEE Access, vol. 9, pp. 96056–96063, doi: https://doi.org/10.1109/ACCESS.2021.3094566
4. Hung, P.T., Truong, M.D.D., Hung, P.D. (2022). Tuning Proximal Policy Optimization Algorithm in Maze Solving with ML-Agents. In: Singh, M., Tyagi, V., (eds) Advances in Computing and Data Sciences. ICACDS 2022. Communications in Computer and Information Science, vol 1614. Springer, doi: https://doi.org/10.1007/978-3-031-12641-3_21
5. Barabash, O., Bandurka, O., Svynchuk, O. & Tverdenko, H. Method of identification of tree species composition of forests on the basis of geographic information database. *Advanced Information Systems*, 2022, vol. 6, no. 4, pp 5-10. DOI: https://doi.org/10.20998/2522-9052.2022.4.01
6. Kuchuk, H. and Malokhvii, E. (2024), "Integration of IOT with Cloud, Fog, and Edge Computing: A Review", *Advanced Information Systems*, vol. *8*(2), pp. 65–78, doi: https://doi.org/10.20998/2522-9052.2024.2.08
7. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778. DOI: https://doi.org/10.1109/CVPR.2016.90
8. Yaloveha, V., Podorozhniak, A. & Kuchuk, H. CNN hyperparameter optimization applied to land cover classification. Radioelectronic and computer systems, 2022, no. 1 (101), pp. 115-128. DOI: https://doi.org/10.32620/reks.2022.1.09
9. Tan, M. & Le, Q. V. Efficientnetv2: Smaller models and faster training. ArXiv (Cornell University), Preprint arXiv:2104.00298, 2021. DOI: https://doi.org/10.48550/arXiv.2104.00298

Received (Надійшла) 11.07.2025
Accepted for publication (Прийнята до друку) 22.10.2025

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Радченко В'ячеслав Олексійович – старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Viacheslav Radchenko – Senior Lecturer of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: viacheslav.radchenko@nure.ua; ORCID Author ID: https://orcid.org/0000-0001-5782-1932;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57189376280.

Андрусенко Юлія Олександрівна – асистентка кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Yuliia Andrusenko – Assistant Lecturer of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: yuliia.andrusenko@nure.ua; ORCID Author ID: https://orcid.org/0000-0001-7844-2042;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=59412400500&origin=resultslist.

Інтелектуальний підхід до планування з урахуванням концепції допустимого балансу робіт

В. О. Радченко, Ю. О. Андрусенко

Анотація. У статті представлено інтелектуальний підхід до задачі планування обчислювальних робіт у розподіленому середовищі з урахуванням концепції допустимого балансу навантаження. Запропонована модель поєднує принципи багатокритеріальної оптимізації та машинного навчання для адаптивного розподілу ресурсів між вузлами системи. Введення коефіцієнта допустимості дозволяє динамічно обмежувати набір машин, доступних для виконання конкретної роботи, що підвищує ефективність використання потужностей та зменшує час очікування задач. Розроблений підхід враховує різні критерії ефективності — продуктивність, баланс навантаження та стабільність планування — і може бути реалізований у грід- та хмарних інфраструктурах. Експериментальні результати демонструють, що застосування інтелектуальних алгоритмів дозволяє досягти оптимального компромісу між швидкодією системи та рівномірністю завантаження обчислювальних ресурсів.

Ключові слова: планування, розподілені системи, ГРІД, ресурси, гетерогенні системи.